Hierarchical Bayesian modeling: an example in reliability analysis

contact: jean-marc.freyermuth@univ-amu.fr

Master DS, Academic year 2022-2023

Section 1

Data analysis problem: modeling approach I

The dataset and the Bayesian hierarchical model

The table hereafter describes the multiple failures of pumps involved in the cooling system of a nuclear power plant. We consider that the number of observed failures y_i of the i-th pump during a time period t_i (in thousands of hours) follow a Poisson distribution $Poi(\lambda_i t_i)$. In order to build a Bayesian model for the number of failures, we specify the following **hierarchical** prior:

$$\begin{aligned} &\lambda_i|\beta \sim \mathcal{G}(\alpha, \beta),\ 1 \leq i \leq 10 \\ &\beta \sim \mathcal{G}(\gamma, \delta) \\ &\perp_{1 \leq i \leq n} \lambda_i|\beta. \end{aligned}$$

with hyperparameters $\alpha = 1.8$, $\gamma = 0.01$ $\delta = 1$.

```
##    pump nb_failures time
## 1     1           5   94
## 2     2           1   16
## 3     3           5   63
## 4     4          14  126
## 5     5           3    5
## 6     6          19   31
## 7     7           1    1
## 8     8           1    1
## 9     9           4    2
## 10   10          22   10
```

Questions

- determine the joint posterior distribution and the full posterior conditional distributions.
- provide 95% credibility intervals for the λ_i's and for β.
- Imaging that these pumps are connected in parallel, so that the cooling system is working if at least one pump is working. Compute the posterior probability that the system will work well more than 10000 hours (Pumps will not be repaired). *Hint: if* $y_i|\lambda_i, t_i \sim Poi(\lambda_i t_i)$ *then we can show that the time until the first failure is distributed accordingly to an exponential distribution with parameter* λ_i.

Reminder (1)

Reminder: the Poisson distribution for modelling count data (data collected over equal area/time region) is given by

$$y|\theta \sim Poi(\theta)$$

$$p(y = k|\theta) = \frac{\theta^k}{k!} e^{-\theta}, \ k = 0, 1, \ldots$$

Reminder (2)

Let us now consider small (time) areas of length $h > 0$. **Assume** that

- if h is small enough, the probability of observing an event on this time interval h is proportional to h

$$P(1 \text{ event within time period of length } h) = \lambda h + o(h).$$

where $o(h)/h \to 0,\ h \to 0$, λ is the intensity of the process.

- the probability of observing more than one event over the time interval h is negligible if h small.

$$P(2 \text{ or more event within } h) = o(h).$$

- events that occur within two disjoint intervals are independent.

Reminder (3)

Under these conditions,

$$y|\lambda, t \sim Poi(\lambda t)$$

$$p(y|\lambda, t) = \frac{(\lambda t)^y}{y!} e^{-\lambda t}, \ k = 0, 1, \ldots$$

Section 2

Modelling approach I: solution

Solution (1)

Likelihood:

$$\begin{cases} y_i|t_i, \lambda_i \sim Poi(\lambda_i t_i) \\ p(y_i|\lambda_i, t_i) = \dfrac{(\lambda_i t_i)^{y_i}}{y_i!} e^{-\lambda_i t_i} \end{cases}$$

Prior:

$$\begin{cases} \lambda_i|\beta \sim \mathcal{G}(\alpha, \beta), \ 1 \leq i \leq 10, \ \perp_{1 \leq i \leq 10} \lambda_i|\beta \\ \pi(\lambda_i|\beta) = \dfrac{\beta^\alpha}{\Gamma(\alpha)} \lambda_i^{\alpha-1} e^{-\beta\lambda_i} \end{cases}$$

$$\begin{cases} \beta \sim \mathcal{G}(\gamma, \delta) \\ \pi(\beta) = \dfrac{\delta^\gamma}{\Gamma(\gamma)} \beta^{\gamma-1} e^{-\beta\delta}, \end{cases}$$

with $\alpha = 1.8$, $\gamma = 0.01$ $\delta = 1$.

Solution (2)

Note that $\pi(\lambda_1, \ldots, \lambda_{10}, \beta) = \pi(\beta)\pi(\lambda_1, \ldots, \lambda_{10}|\beta)$. Hence the full posterior distribution is given by

$$\pi(\lambda_{1:10}, \beta|D) \propto \pi(\beta) \prod_{i=1}^{10} p(y_i|t_i, \lambda_i, \beta)\pi(\lambda_i|\beta)$$

$$\propto \beta^{10\alpha+\gamma-1} e^{-\delta\beta} \left(\prod_{i=1}^{10} \lambda_i^{y_i+\alpha-1} e^{-\lambda_i(t_i+\beta)} \right).$$

Solution (3)

The joint posterior distribution is given by:

$$\pi(\lambda_{1:10}, \beta | D) \propto \beta^{10\alpha+\gamma-1} e^{-\delta\beta} \left(\prod_{i=1}^{10} \lambda_i^{y_i+\alpha-1} e^{-\lambda_i(t_i+\beta)} \right).$$

Identification of the full conditional posterior distributions of λ_i's:

$$\begin{cases} \pi(\lambda_i | \lambda_{-i}, \beta, D) \propto \lambda_i^{y_i+\alpha-1} e^{-\lambda_i(t_i+\beta)} \\ \quad \lambda_i | \lambda_{-i}, \beta, D \sim \mathcal{G}(y_i + \alpha, t_i + \beta). \end{cases}$$

Solution (4)

The joint posterior distribution is given by:

$$\pi(\lambda_{1:10}, \beta | D) \propto \beta^{10\alpha+\gamma-1} e^{-\delta\beta} \left(\prod_{i=1}^{10} \lambda_i^{y_i+\alpha-1} e^{-\lambda_i(t_i+\beta)} \right).$$

Identification of the full conditional posterior distributions of β:

$$\begin{cases} \pi(\beta | \lambda_{1:10}, D) \propto \beta^{10\alpha+\gamma-1} e^{-\beta(\delta+\sum_{i=1}^{10} \lambda_i)} \\ \\ \beta | \lambda_{1:10}, D \sim \mathcal{G}(10\alpha + \gamma, \delta + \sum_{i=1}^{10} \lambda_i). \end{cases}$$

Solution (5)

Since we were able to identify the *full posterior conditional distributions*, we can implement the **gibbs** algorithm as follows

- set starting value for $\beta^{(0)}$
- for iteration $m = 1, \ldots, M$
- draw $\lambda_i^{(m)}$'s simultaneously from there full conditional (depending on $\beta^{(m-1)}$)
- draw $\beta^{(m)}$ from its full conditional using $\lambda_i^{(m)}$'s.

Solution (6)

```
number = 1:10
y <- c(5, 1, 5, 14, 3, 19, 1, 1, 4, 22)
t <- c(94, 16, 63, 126, 5, 31, 1, 1, 2, 10)
n = 10
data = cbind(number,y,t)
colnames(data) = c("pump", "failures", "time")
data
```

```
##       pump failures time
##  [1,]    1        5   94
##  [2,]    2        1   16
##  [3,]    3        5   63
##  [4,]    4       14  126
##  [5,]    5        3    5
##  [6,]    6       19   31
##  [7,]    7        1    1
##  [8,]    8        1    1
##  [9,]    9        4    2
## [10,]   10       22   10
```

Solution (6)

```
# number of monte carlo iteration
M = 20000
# specification of the hyperparameters
alpha = 1.8;delta = 1;gamma = 0.01;burnin = 500

# intialization of parameter values
beta = rep(0, length = (M+burnin))
beta[1] = 1
lambda = matrix(nrow = (M+burnin), ncol = 10,0)

for (m in 2:(M+burnin)) {
  lambda[m, ] = rgamma(n, y + alpha, t + beta[m-1])
  beta[m] = rgamma(1, n * alpha + gamma, delta + sum(lambda[m, ]))
}
lambda = lambda[(burnin+1):(M+burnin), ]
beta = beta[(burnin+1):(M+burnin)]

# summary of the posterior
mean(beta)
```

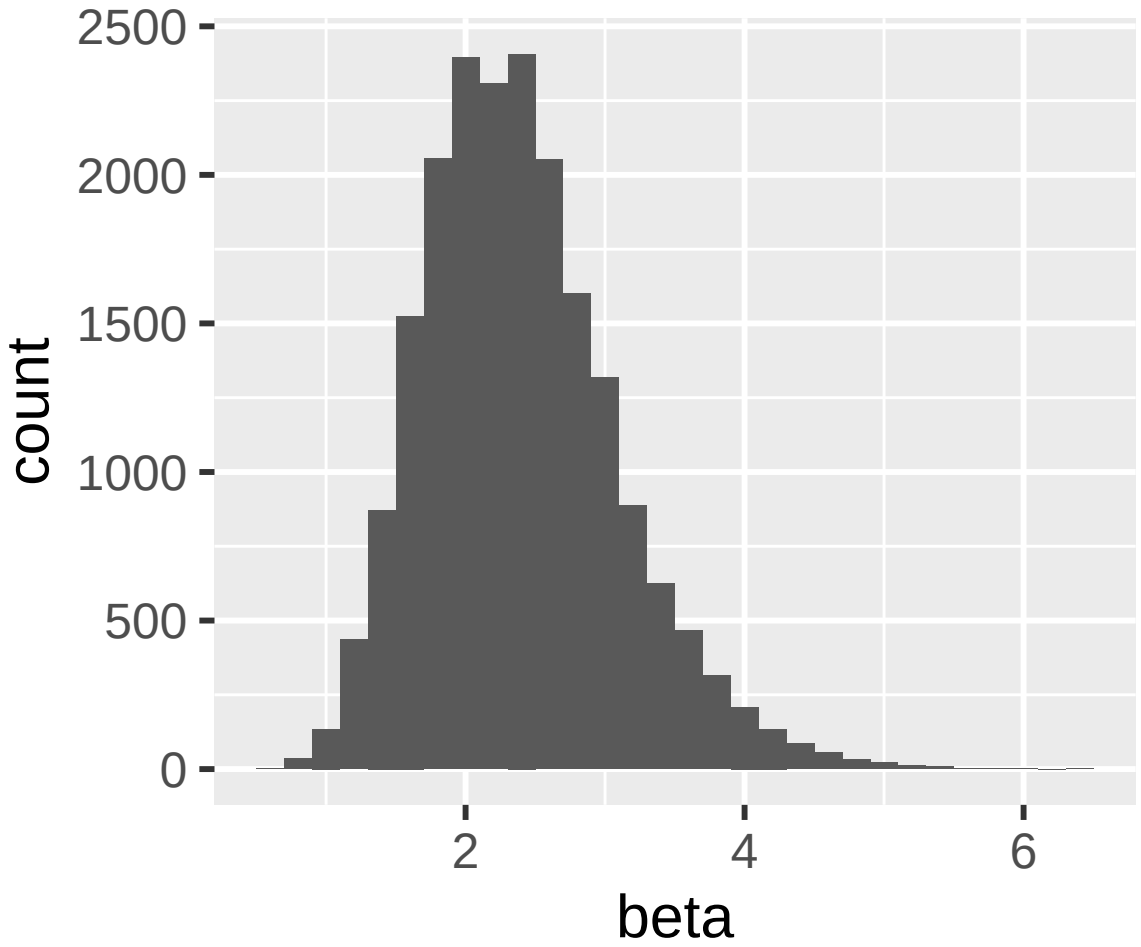
```
## [1] 2.389047
```

```
colMeans(lambda)
```

Solution (7)

```
ggplot(data.frame(beta=beta), aes(x=beta)) +
  geom_histogram(position="identity")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Solution (8)

reliability of the system

Let $R_S(t_F)$ be the reliability of a system of pumps in parallel, i.e, the probability that the system is working till the time t_F. Since the time to the first failure of the i-th pump is $t_i|\lambda_i \sim Exp(\lambda_i)$. We readily deduce that

$$R_S(t_F) = 1 - \prod_{i=1}^{10}(1 - \exp(-t_F\lambda_i)).$$

Since we have generated chains from the posterior distribution of the λ's, we can easily get the posterior distribution of $R_S(t_F)$ and therefore we can compute the posterior probability that the system will work more than 10000 hours.

```
post_Rf = 1-apply((1-exp(-10*lambda[,1:10 ])), 1, function(x){prod(x)})
mean(post_Rf)
```

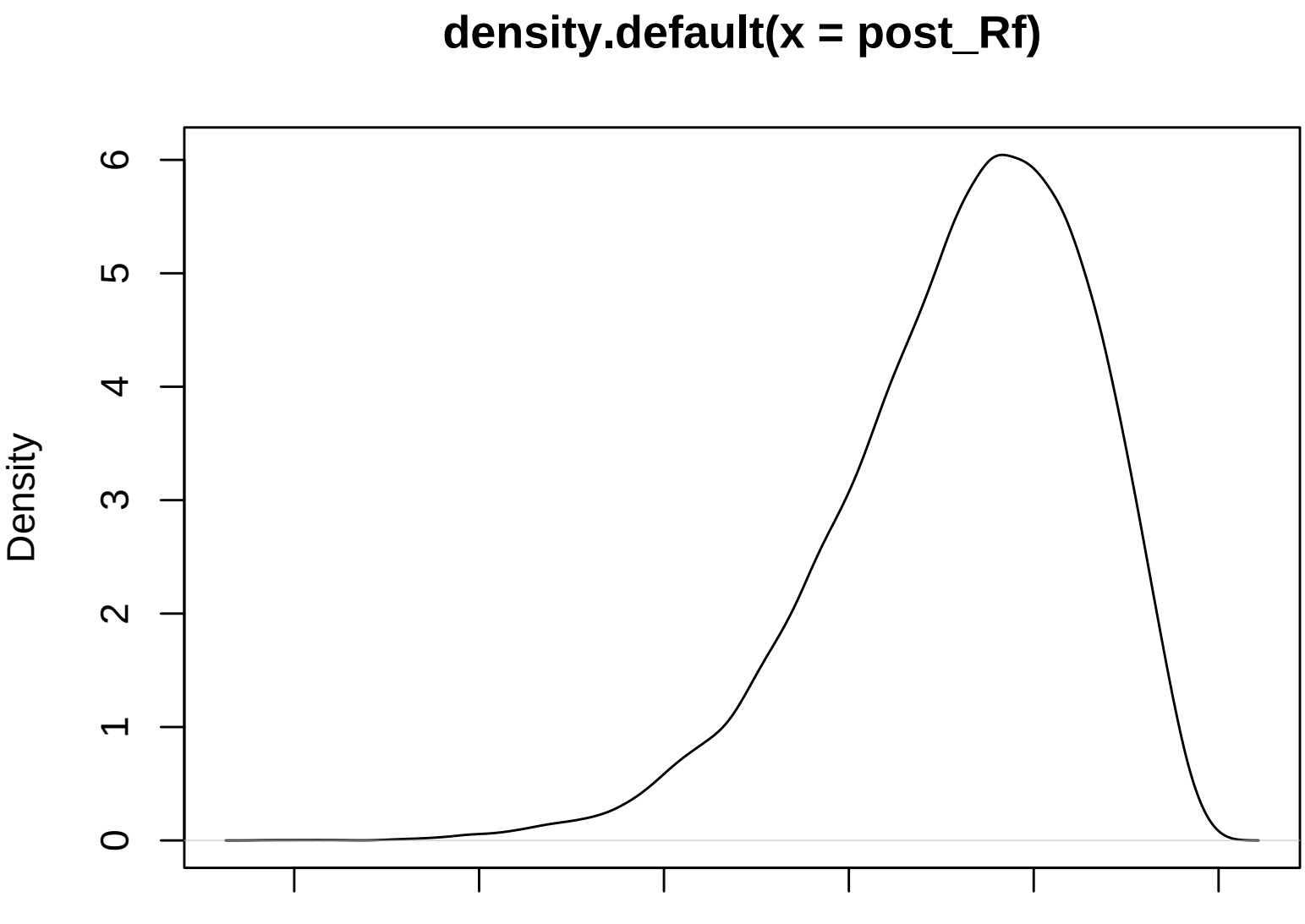
```
## [1] 0.8611587
```

```
quantile(post_Rf, c(0.025,0.975))
```

```
##      2.5%     97.5%
## 0.7042977 0.9664389
```

Solution (9)

Posterior distribution of $R_s(10)$



Solution (10)

using RJags

R libraries

```
library(rjags, quietly = TRUE)
```

```
## Warning: le package 'rjags' a été compilé avec la version R 4.1.3
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```
library(R2jags, quietly = TRUE)
```

```
##
## Attachement du package : 'R2jags'
```

```
## L'objet suivant est masqué depuis 'package:coda':
##
##     traceplot
```

```
library(coda)
```

Solution (11)

```
model_code = '
model
{
  for (i in 1:N)
  {
    lambda[i] ~ dgamma(1.8, beta)
    tau[i] <-  (lambda[i]*t[i])
    y[i] ~ dpois(tau[i])
  }
beta  ~ dgamma(0.01,1)
}
'
model_data = list(y = c(5, 1, 5, 14, 3, 19, 1, 1, 4, 22),
t = c(94, 16, 63, 126, 5, 31, 1, 1, 2, 10),
N  = 10)
```

Solution (12)

```
model_parameters =  c("beta","lambda")
# Run the model
model_run = jags(data = model_data,
                 parameters.to.save = model_parameters,
                 model.file=textConnection(model_code),
                 n.chains=4,
                 n.iter=10000,
                 n.burnin=200,
                 n.thin=2)
```

```
## module glm loaded

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 11
##    Total graph size: 45
##
## Initializing model
```

Solution (13)

```
print(model_run)
```

```
## Inference for Bugs model at "4", fit using jags,
##  4 chains, each with 10000 iterations (first 200 discarded), n.thin = 2
##  n.sims = 19600 iterations saved
##             mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
## beta          2.393   0.691  1.288  1.898  2.310  2.802  3.967 1.001 20000
## lambda[1]     0.071   0.027  0.028  0.051  0.067  0.087  0.133 1.001 20000
## lambda[2]     0.153   0.092  0.030  0.086  0.136  0.202  0.383 1.001 20000
## lambda[3]     0.104   0.040  0.041  0.075  0.099  0.128  0.196 1.001 12000
## lambda[4]     0.123   0.031  0.070  0.101  0.121  0.142  0.189 1.001 20000
## lambda[5]     0.658   0.307  0.208  0.435  0.610  0.831  1.380 1.001  7900
## lambda[6]     0.624   0.139  0.382  0.526  0.614  0.711  0.927 1.001 20000
## lambda[7]     0.855   0.548  0.153  0.462  0.736  1.124  2.231 1.001  9800
## lambda[8]     0.854   0.547  0.158  0.459  0.737  1.119  2.195 1.001 12000
## lambda[9]     1.347   0.605  0.450  0.909  1.253  1.681  2.769 1.001 12000
## lambda[10]    1.928   0.407  1.219  1.639  1.899  2.186  2.812 1.001 20000
## deviance     43.561   4.501 36.723 40.283 42.903 46.113 53.989 1.001 20000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 10.1 and DIC = 53.7
```

Section 3

Modeling approach II

Questions

Answer the same question but you now consider the following prior:

$$\begin{aligned}
&\lambda_i|\alpha,\beta \sim \mathcal{G}(\alpha,\beta),\ 1 \leq i \leq 10\\
&\beta \sim \mathcal{G}(\gamma,\delta)\\
&\alpha \sim \mathcal{U}(0,0.5)\\
&\perp_{1\leq i\leq n} \lambda_i|\alpha,\beta.
\end{aligned}$$

with $\gamma = 0.01$ $\delta = 1$.

Solution (1)

$$\begin{cases} y_i|t_i,\lambda_i \sim Poi(\lambda_i t_i) \\ p(y_i|\lambda_i,t_i) = \dfrac{(\lambda_i t_i)^{y_i}}{y_i!} e^{-\lambda_i t_i}, \end{cases}$$

$$\begin{cases} \lambda_i|\alpha,\beta \sim \mathcal{G}(\alpha,\beta),\ 1 \leq i \leq 10 \\ \pi(\lambda_i|\alpha,\beta) = \dfrac{\beta^\alpha}{\Gamma(\alpha)} \lambda_i^{\alpha-1} e^{-\beta\lambda_i}, \end{cases}$$

$$\begin{cases} \beta \sim \mathcal{G}(\gamma,\delta) \\ \pi(\beta) = \dfrac{\delta^\gamma}{\Gamma(\gamma)} \beta^{\gamma-1} e^{-\beta\delta}, \end{cases}$$

$$\begin{cases} \alpha \sim \mathcal{U}(0,5) \\ \pi(\alpha) = 1_{(0,5]}(\alpha), \end{cases}$$

with $\gamma = 0.01$ $\delta = 1$. Assumption on prior $\perp_{1\leq i\leq n} \lambda_i|\alpha,\beta$.

Solution (2)

$$\begin{aligned}
\pi(\lambda_{1:10}, \alpha, \beta | D) &\propto \pi(\lambda_{1:10}, \alpha, \beta) p(D | \lambda_{1:10}, \alpha, \beta) \\
&\propto \prod_{i=1}^{10} \left[p(y_i | t_i, \lambda_i, \alpha, \beta) \right] \pi(\alpha) \pi(\beta) \prod_{i=1}^{10} \pi(\lambda_i | \beta) \\
&\propto \beta^{10\alpha+\gamma-1} e^{-\delta\beta} 1_{(0,5]}(\alpha) \Gamma(\alpha)^{-10} \left(\prod_{i=1}^{10} \lambda_i^{y_i+\alpha-1} e^{-\lambda_i(t_i+\beta)} \right)
\end{aligned}$$

Solution (3)

Determine the full conditional posterior distribution

$$\pi(\lambda_i|\lambda_{-i},\alpha,\beta,D) \propto \quad \lambda_i^{y_i+\alpha-1} e^{-\lambda_i(t_i+\beta)}$$
$$\lambda_i|\lambda_{-1},\beta,D \sim \quad \mathcal{G}(y_i+\alpha, t_i+\beta)$$

Solution (4)

$$\pi(\beta|\lambda_{1:10}, \alpha, D) \propto \beta^{10\alpha+\gamma-1} e^{-\beta(\delta+\sum_{i=1}^{10}\lambda_i)}$$

$$\beta|\lambda_{1:10}, D \sim \mathcal{G}(10\alpha+\gamma, \delta+\sum_{i=1}^{10}\lambda_i)$$

$$\pi(\alpha|\lambda_{1:10}, \beta, D) \propto 1_{(0,5]}(\alpha)(\Gamma(\alpha))^{-10}\beta^{10\alpha+\delta-1}\left(\prod_{i=1}^{10}\lambda_i^{y_i+\alpha-1}\right)$$

$$\alpha|\lambda_{1:10}, \beta, D \sim ?$$

As such, we cannot use the gibbs sampler... we need other MCMC tools

Section 4

Metropolis algortihm in a nutshell

Accept-Reject algorithm

We aim at generating sample from a posterior density $\theta|D \sim f$ with known pdf up to a multiplicative constant.

If $g(.)$ is a density from which we can easily drawn samples and that there exists a constant $C > 0$ such that $\frac{f(\theta)}{Cg(\theta)} \leq 1, \ \forall \theta$.Then

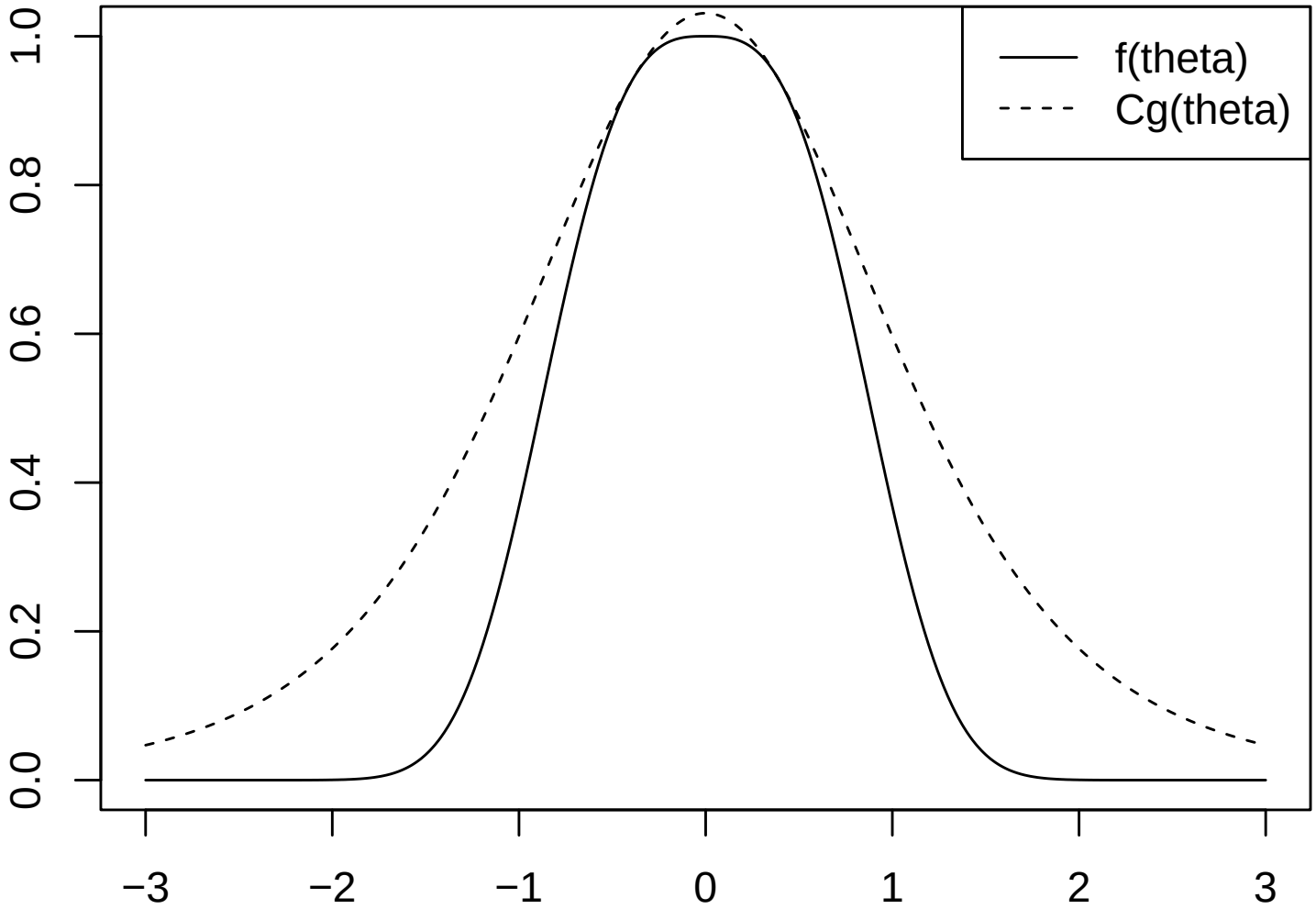
- draw a sample θ_{prop} from $g(\theta)$
- accept θ_{prop} with a probabiliy $\frac{f(\theta_{prop})}{Cg(\theta_{prop})}$.

Accept-Reject algorithm

Imagine your posterior density is of the form $\propto e^{-|\theta|^3}$.

```
f = function(theta) {exp(-abs(theta)^3)};g= function(theta) {dt(theta, df = 5)}
rg = function(n) {rt(n=n,df=5)};

z= seq(-3,3,length = 1000);C= max(f(z)/g(z));

M = 10000; theta = rep(0, length = M); iter = 0; n_accepted = 0;

while (n_accepted <M)
{
  iter = iter + 1;
  theta_prop = rg(1);
  p_accepted = f(theta_prop)/ (C * g(theta_prop))

  if (runif(1) <= p_accepted)
  {
    n_accepted = n_accepted + 1;
    theta[n_accepted] = theta_prop;
  }
}

cat("taux d'acceptation", round(M/iter,2), "\n")
```

Accept-Reject algorithm



The Metropolis algorithm

This algorithm generates a random walk where the updates (candidate values) are accepted or rejected accordingly to a proper rule

- *step 0:* choose a starting value for $\theta^{(0)}$
- *step 1:* at iteration m, draw θ^{prop} from a **symmetric proposal distribution** $q_m(\theta^{prop}|\theta^{(m-1)})$
- *step 2:* compute: $Prob = \min\left\{1, \frac{p(\theta^{prop}|x)}{p(\theta^{(m-1)}|x)}\right\}$
- *step 3:* set $\theta^{(m)} = \theta^{prop}$ with probability $Prob$ and $\theta^{(m)} = \theta^{(m-1)}$ otherwise. Go back to step 1.

remark: the generalization of this algorithm known as Metropolis-Hasting uses asymmetric proposal distributions.

Example of Metropolis algorithm

```
log_f = function(theta) - abs(theta)^3;

MH = function(log_f, M, sd_prop, init)
{
  theta = rep(0, length = M);
  n_accepted = 0;
  theta[1] = init;
  for (m in 2:M)
  {
    theta_prop = theta[m-1] + rnorm(1, 0, sd_prop);
    prob = min(1, exp(log_f(theta_prop)-log_f(theta[m-1])));

    if (runif(1) <= prob)
    {
      n_accepted = n_accepted  + 1;
      theta[m] = theta_prop;
    }
    else theta[m] = theta[m-1]
  }
  return(list(theta = theta, n_accepted = n_accepted))
}
```

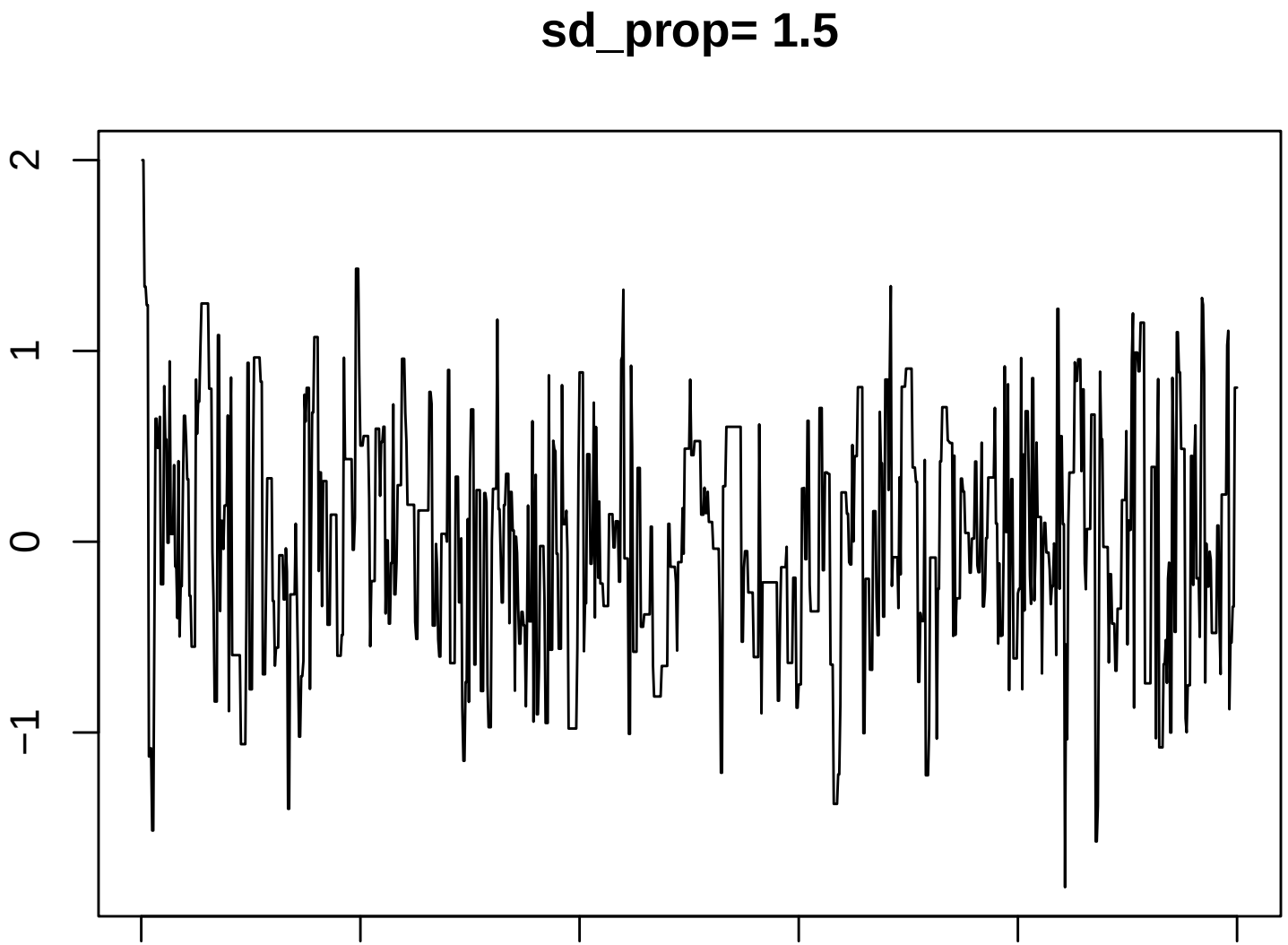
Example of Metropolis algorithm

```
res_MH = MH(log_f, M = 10000, sd_prop = 1.5, init = 2)
cat("Taux d'acceptation", round(res_MH$n_accepted/(M-1),2))
```

```
## Taux d'acceptation 0.46
```

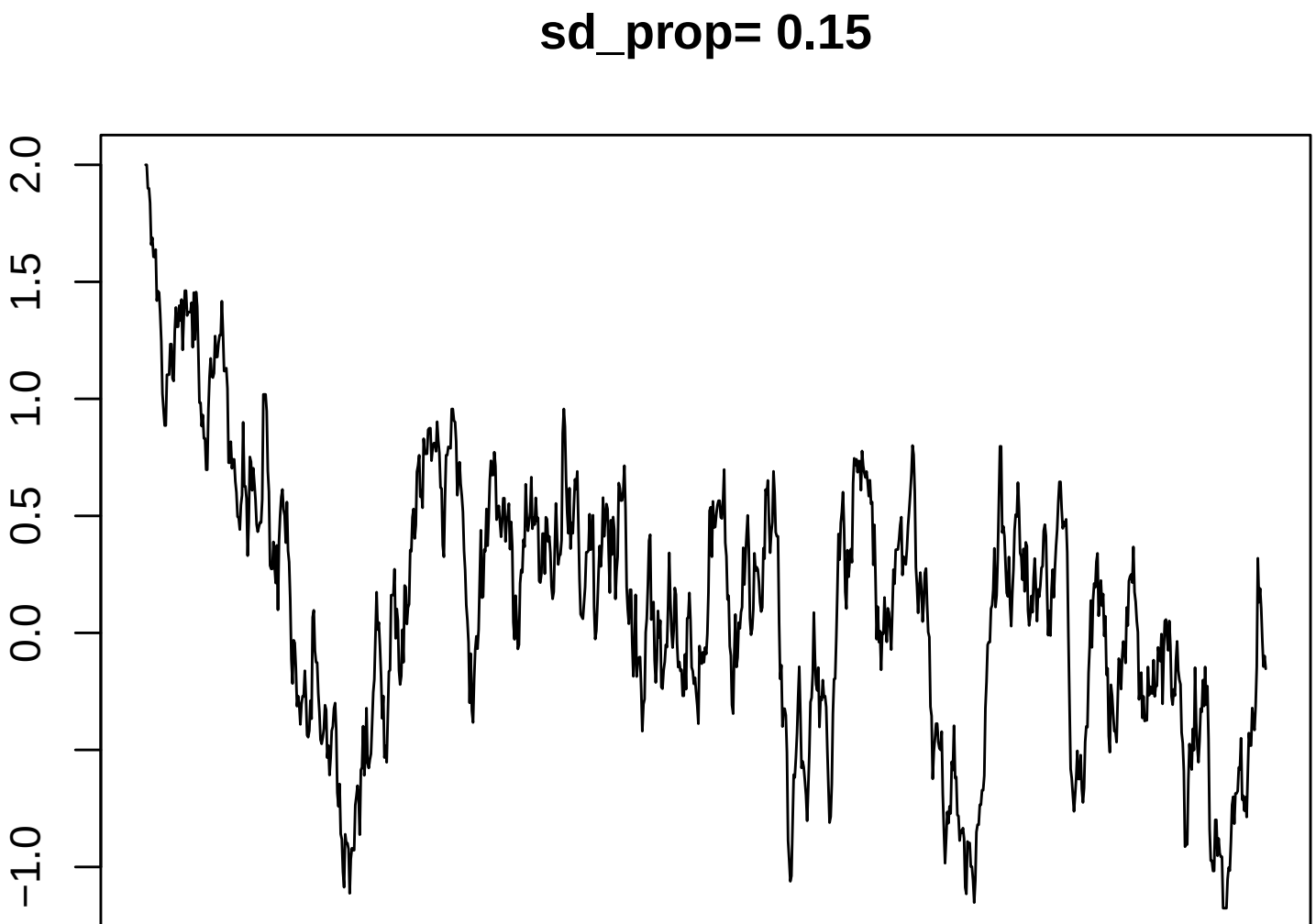
Example of Metropolis algorithm

```
plot(res_MH$theta[1:1000], type = 'l', main = "sd_prop= 1.5", ylab="", xlab="")
```



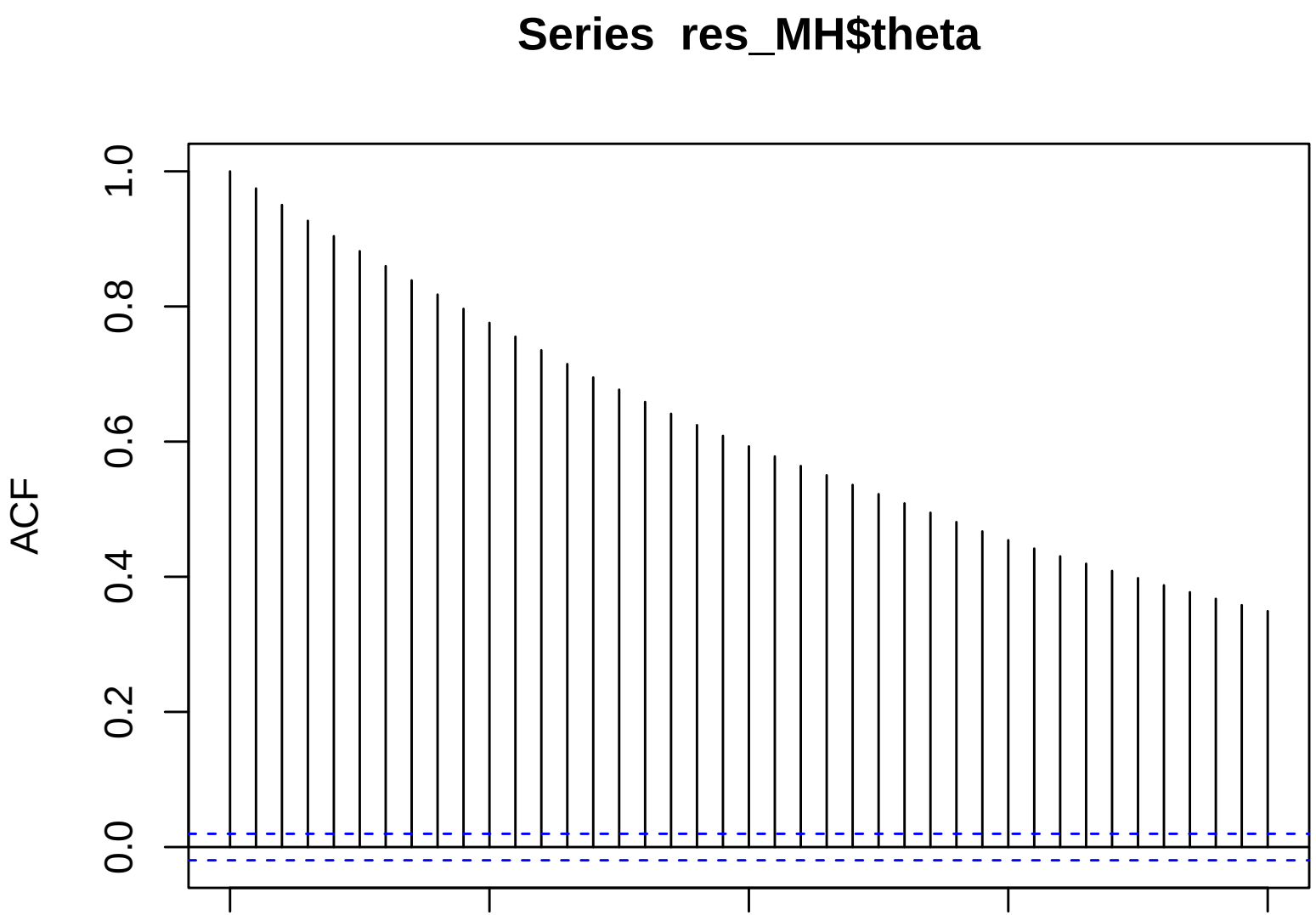
Example of Metropolis algorithm: Mixing of the chain

```
res_MH = MH(log_f, M = 10000, sd_prop = 0.15, init = 2)
plot(res_MH$theta[1:1000], type = 'l', main = "sd_prop= 0.15", ylab="", xlab="")
```



Example of Metropolis algorithm: Mixing of the chain

```
acf(res_MH$theta)
```



Section 5

Modeling approach II solution continued

Solution (5)

Computation using **Metropolis within gibbs**

Step 0: set starting value for $\beta^{(0)}$

Step 1: for iteration $m = 1, \ldots, M$

- draw $\lambda_i^{(m)}$'s simultaneously from there full conditional (depending on $\beta^{(m-1)}, \alpha^{(m-1)}$)
- draw $\beta^{(m)}$ from its full conditional using $\lambda_i^{(m)}$'s and $\alpha^{(m-1)}$.
- Random Walk metropolis step:
  - proposal

$$\alpha_{prop} = \alpha^{(m-1)} + \varepsilon, \ \varepsilon \sim \mathcal{N}(0, \sigma^2_{prop})$$

`+ probability of acceptance`

$$Prob = \min \left\{ 1, \frac{\pi(\alpha_{prop} | \lambda_{1:10}^{(m)}, \beta^{(m)}, Data)}{\pi(\alpha^{(m-1)} | \lambda_{1:10}^{(m)}, \beta^{(m)}, Data)} \right\}$$

Step 2: go back to Step 1; iterate.

Solution (6)

```
log_posterior= function(lambda, alpha, beta, supp_alpha){
  if ((alpha >supp_alpha[1]) & (alpha <supp_alpha[2]))
  {
      (alpha) * sum(log(lambda))  +
      10 * alpha * log(beta) - 10  * log(gamma(alpha))
  } else
  {
    -1e12
  }
}
M = 20000
names_theta =  c(LETTERS[seq( from = 1, to = 10 )], "alpha", "beta")
```

Solution (7)

```
MCMC_pumps = function(M, y, t, sd_prop, beta_start,
        alpha_start, lambda_start, names_theta, supp_alpha)
{
  length.y = 10; theta_post = matrix(nrow = M, ncol = 12, 0)
  theta_post[1, ] = c(lambda_start,alpha_start, beta_start)
  colnames(theta_post) =names_theta;delta = 1;  gam = 0.01;  n.accept = 0;
  for (i in 2:M){
  theta_post[i, 1:10 ] = rgamma(length.y, y + theta_post[(i-1), 11],
                                          t + theta_post[(i-1), 12])
  theta_post[i, 12] = rgamma(1, 10*theta_post[(i-1), 11] + gam, delta
                                            + sum(theta_post[i, 1:10]))
  alpha_prop = theta_post[(i-1), 11] +  rnorm(1, mean = 0, sd = sd_prop)
  prob = min(1,exp(log_posterior(theta_post[i,1:10], alpha_prop,
                                        theta_post[i, 12], supp_alpha)
        -log_posterior(theta_post[i,1:10],theta_post[(i-1), 11],
                                          theta_post[i, 12],supp_alpha)))
  accept = ((runif(1)<= prob))
  if (accept == TRUE){n.accept = n.accept + 1; theta_post[i, 11]= alpha_prop
  } else
  { theta_post[i, 11] = theta_post[(i-1), 11]
  }
}
accept.rate = n.accept / M
  return(list(accept.rate = accept.rate, theta_post = theta_post))
```

Solution (8)

```
alpha_start = 0.1
beta_start = 1
supp_alpha = c(0,5)
lambda_start = rep(1, length = 10)
sd_prop = sqrt(0.250)
obj_mcmc = MCMC_pumps(M, y, t, sd_prop, beta_start, alpha_start, lambda_start, names
theta_post = obj_mcmc$theta_post
apply(theta_post[2000:M,], 2, mean)
```

```
##          A          B          C          D          E          F          G
## 0.06035176 0.10309970 0.08981490 0.11636450 0.63122895 0.61704730 0.93737304
##          H          I          J      alpha       beta
## 0.93408581 1.64803103 2.07464762 0.75228569 0.95736144
```

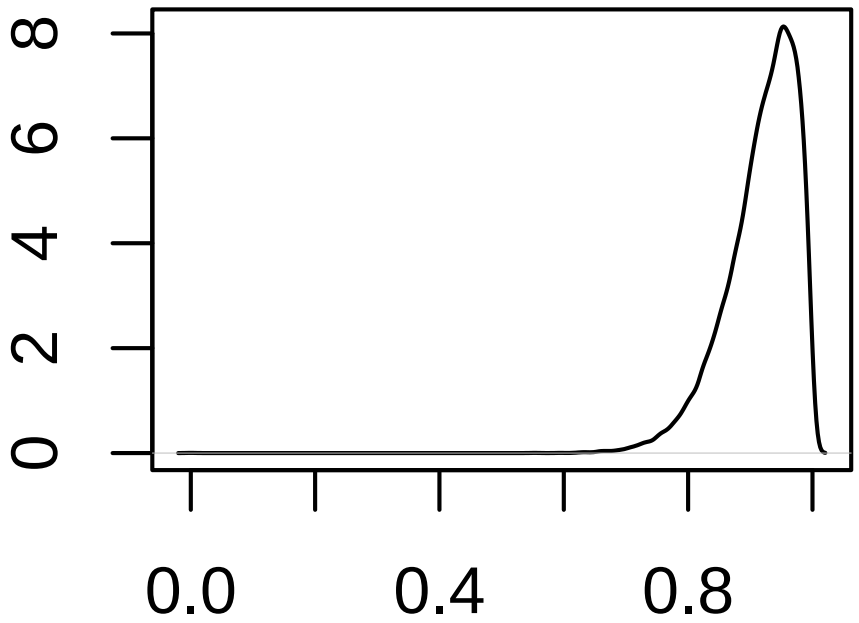
```
obj_mcmc$accept.rate
```

```
## [1] 0.40525
```

Solution (9)

Distribution of the posterior probability that the system will survive 10000 hours

```
post_Rf = 1-apply((1-exp(-10*theta_post[, 1:10])), 1, function(x){prod(x)})
plot(density(post_Rf), main ="", xlab="",ylab="")
```



Solution (10)

```
library("coda")
hpd <- HPDinterval(as.mcmc(theta_post), 0.95)  ###highest posterior density interval
hpd
```

```
##                lower     upper
## A      0.0175602279 0.1105380
## B      0.0005704837 0.2596448
## C      0.0246052798 0.1650441
## D      0.0599085798 0.1759306
## E      0.1001164977 1.2786518
## F      0.3576219063 0.8929171
## G      0.0012471762 2.3988352
## H      0.0006694333 2.3826908
## I      0.3435319578 3.2325718
## J      1.2449834852 2.9546306
## alpha  0.2506975604 1.3259634
## beta   0.0828871561 2.0412196
## attr(,"Probability")
## [1] 0.95
```

Solution (11)

Using Rjags

R program:

```
model_code = '
model
{
  for (i in 1:N)
  {
    lambda[i] ~ dgamma(alpha, beta)
    tau[i] <-  (lambda[i]*t[i])
    y[i] ~ dpois(tau[i])
  }
beta  ~ dgamma(0.001,0.001)
alpha  ~ dunif(0.001,5)
}
'
model_data = list(y = c(5, 1, 5, 14, 3, 19, 1, 1, 4, 22),
t = c(94, 16, 63, 126, 5, 31, 1, 1, 2, 10),
N  = 10)
```

Solution (12)

```
model_parameters =  c("alpha","beta","lambda")
# Run the model
model_run = jags(data = model_data,
                 parameters.to.save = model_parameters,
                 model.file=textConnection(model_code),
                 n.chains=4,
                 n.iter=10000,
                 n.burnin=200,
                 n.thin=2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 12
##    Total graph size: 45
##
## Initializing model
```

Solution (13)

```
print(model_run)
```

```
## Inference for Bugs model at "5", fit using jags,
##  4 chains, each with 10000 iterations (first 200 discarded), n.thin = 2
##  n.sims = 19600 iterations saved
##             mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
## alpha         0.949   0.413  0.360  0.654  0.877  1.164  1.938 1.001 20000
## beta          1.501   0.961  0.292  0.826  1.290  1.941  3.911 1.001  6200
## lambda[1]     0.062   0.026  0.022  0.043  0.059  0.078  0.121 1.001 20000
## lambda[2]     0.110   0.081  0.011  0.050  0.091  0.150  0.316 1.001 12000
## lambda[3]     0.092   0.038  0.033  0.064  0.087  0.114  0.182 1.001  7200
## lambda[4]     0.118   0.030  0.066  0.096  0.115  0.137  0.184 1.001  8900
## lambda[5]     0.612   0.317  0.159  0.381  0.558  0.786  1.383 1.001  9900
## lambda[6]     0.614   0.138  0.374  0.516  0.603  0.700  0.916 1.001  4300
## lambda[7]     0.833   0.650  0.083  0.370  0.669  1.113  2.547 1.001 12000
## lambda[8]     0.839   0.662  0.085  0.369  0.676  1.120  2.542 1.001  9900
## lambda[9]     1.480   0.737  0.440  0.945  1.352  1.871  3.238 1.001 20000
## lambda[10]    2.007   0.438  1.249  1.692  1.979  2.280  2.955 1.001 13000
## deviance     43.149   4.353 36.660 39.957 42.480 45.634 53.252 1.001 20000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
```